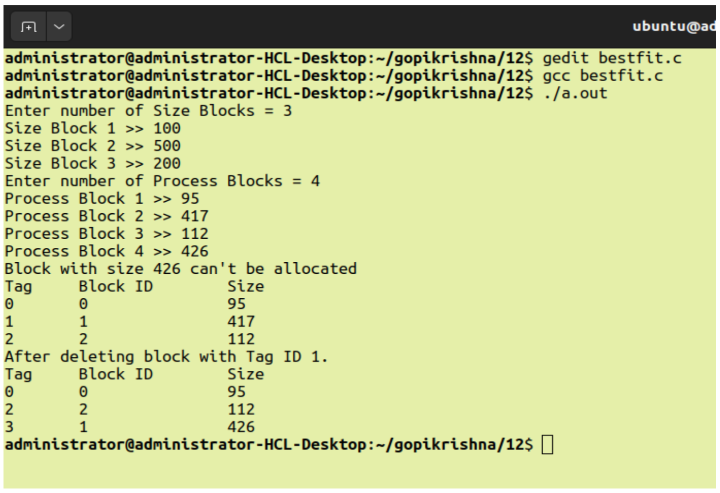
```
/*
            GOPIKRISHNA V
            S3 CSE A
            52
*/
#include<stdio.h>
#include<stdlib.h>

int g = 0, k = 0;

struct free
{
   int tag;
   int size;
   struct free* next;
}*free_head = NULL,*prev_free = NULL;

struct alloc
{
   int block_id;
   int tag;
   int size;
   struct alloc* next;
}*alloc_head = NULL,*prev_alloc = NULL;

void create_free(int c)
{
   struct free* p = (struct free*)malloc(sizeof(struct free));
   p->size = c;
   p->tag = g;
   p->next = NULL;
   if (free_head == NULL)
      free_head = p;
   else
      prev_free->next = p;
   prev_free = p;
   g++;
}

void print_free()
{
   struct free* p = free_head;
   printf("Tag\tSize\n");
   while (p != NULL)
   {
      printf("%d <> %d\n",p->tag,p->size);
      p = p->next;
   }
```

```
}

void print_alloc()
{
   struct alloc* p = alloc_head;
   printf("Tag\tBlock ID\tSize\n");
   while (p != NULL)
   {
      printf("%d\t%d\t\t%d\n",p->tag,p->block_id,p->size);
      p = p->next;
   }
}

void create_alloc(int c)
{
   struct alloc* q = (struct alloc*)malloc(sizeof(struct alloc));
   q->size = c;
   q->tag = k;
   q->next = NULL;
   struct free* p = free_head;

   struct free* r = (struct free*)malloc(sizeof(struct free));
   r->size = 99999;

   while (p != NULL)
   {
      if (q->size <= p->size)
      {
         if (p->size < r->size)
            r = p;
      }
      p = p->next;
   }

   if (r->size != 99999)
   {
      q->block_id = r->tag;
      r->size = q->size;
      if (alloc_head == NULL)
         alloc_head = q;
      else
      {
         prev_alloc = alloc_head;
         while (prev_alloc->next != NULL)
            prev_alloc = prev_alloc->next;
         prev_alloc->next = q;
      }
      k++;
```

```
    }
    else
        printf("Block with size %d can't be allocated\n",c);
}

void delete_alloc(int t)
{
    struct alloc *p = alloc_head, *q = NULL;
    while (p != NULL)
    {
        if (p->tag == t)
            break;
        q = p;
        p = p->next;
    }
    if (p == NULL)
        printf("Tag ID doesn't exist\n");
    else
    {
        if (p == alloc_head)
            alloc_head = alloc_head->next;
        else
            q->next = p->next;
    }
    struct free* temp = free_head;
    while (temp != NULL)
    {
        if (temp->tag == p->block_id)
        {
            temp->size += p->size;
            break;
        }
        temp = temp->next;
    }
}

void main()
{
    int lim;

    printf("Enter number of Size Blocks = ");
    scanf("%d",&lim);
    int blockSize[lim];
    for (int i = 0; i < lim; i++)
    {
        printf("Size Block %d >> ",i+1);
        scanf("%d",&blockSize[i]);
    }
    printf("Enter number of Process Blocks = ");
    scanf("%d",&lim);
    int processSize[lim];
    for (int i = 0; i < lim; i++)
    {
        printf("Process Block %d >> ",i+1);
        scanf("%d",&processSize[i]);
    }

    int m = sizeof(blockSize)/ sizeof(blockSize[0]);
    int n = sizeof(processSize)/ sizeof(processSize[0]);

    for (int i = 0; i < m; i++)
        create_free(blockSize[i]);

    for (int i = 0; i < n; i++)
        create_alloc(processSize[i]);
    print_alloc();

    delete_alloc(1);

    create_alloc(426);
    printf("After deleting block with Tag ID 1.\n");
    print_alloc();
}
```

OUTPUT



```
administrator@administrator-HCL-Desktop:~/gopikrishna/12$ gedit bestfit.c
administrator@administrator-HCL-Desktop:~/gopikrishna/12$ gcc bestfit.c
administrator@administrator-HCL-Desktop:~/gopikrishna/12$ ./a.out
Enter number of Size Blocks = 3
Size Block 1 >> 100
Size Block 2 >> 500
Size Block 3 >> 200
Enter number of Process Blocks = 4
Process Block 1 >> 95
Process Block 2 >> 417
Process Block 3 >> 112
Process Block 4 >> 426
Block with size 426 can't be allocated
Tag     Block ID        Size
0       0               95
1       1               417
2       2               112
After deleting block with Tag ID 1.
Tag     Block ID        Size
0       0               95
2       2               112
3       1               426
administrator@administrator-HCL-Desktop:~/gopikrishna/12$
```